Computers, Materials & Continua CMC, vol.62, no.3, pp.1025-1051, 2020

Hybrid Clustering Algorithms with GRASP to Construct an Initial Solution for the MVPPDP

Abeer I. Alhujaylan[1, 2, *] and Manar I. Hosny[1]

Abstract: Mobile commerce (m-commerce) contributes to increasing the popularity of electronic commerce (e-commerce), allowing anybody to sell or buy goods using a mobile device or tablet anywhere and at any time. As demand for e-commerce increases tremendously, the pressure on delivery companies increases to organise their transportation plans to achieve profits and customer satisfaction. One important planning problem in this domain is the multi-vehicle profitable pickup and delivery problem (MVPPDP), where a selected set of pickup and delivery customers need to be served within certain allowed trip time. In this paper, we proposed hybrid clustering algorithms with the greedy randomised adaptive search procedure (GRASP) to construct an initial solution for the MVPPDP. Our approaches first cluster the search space in order to reduce its dimensionality, then use GRASP to build routes for each cluster. We compared our results with state-of-the-art construction heuristics that have been used to construct initial solutions to this problem. Experimental results show that our proposed algorithms contribute to achieving excellent performance in terms of both quality of solutions and processing time.

Keywords: Multi-vehicle profitable pickup and delivery problem, K-means clustering algorithm, ant colony optimisation, greedy randomised adaptive search procedure, metaheuristic algorithms.

1 Introduction

Today, most people rely on electronic commerce (e-commerce) to meet their needs. E-commerce has several advantages that encourage people to rely on it rather than traditional purchasing methods, such as availability of products at any time, low prices, privacy, comfort, and quality assurance. In addition to those advantages, the spread of mobile commerce (m-commerce) has increased the popularity of e-commerce, allowing people to sell or buy goods using a mobile device or tablet anywhere and at any time. As the demand for e-commerce has increased, the pressure on delivery companies to organise their transportation plans to achieve profits and customer satisfaction has also increased.

[1] Computer Science Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia.

[2] College of Computer, Qassim University, Buraydah, Saudi Arabia.

[*] Corresponding Author: Abeer I. Alhujaylan. Email: abeer.alhujaylan@gmail.com.

Transportation means used by millions of people to transport themselves or their goods all over the world make the transport and shipment sector a hotspot in the research field. Most of this research aims to optimise the planning and organisation of the different transportation means used: land, air, and sea. Land transport is considered the most important, because it is used daily in different countries and has a huge volume that includes buses, trucks, cars, trains, trams, motorcycles, and more.

In addition, the increasingly harmful impact of land transport, encourages researchers in different fields (operations research, computer science, and industrial engineering) to find the best solutions possible. There are two negative impacts of land transport: environmental and economic effects. According to environmental research, transport has a main role in distributing pollution and increasing global warming, which results in many diseases that have negative effects on people's health. Therefore, researchers aim to find solutions to decrease congestion and the environmental damage caused by harmful emissions of greenhouse gases and carbon dioxide. On the other hand, the economic effects on companies that work in the transport and shipment sector includes miss-organisation of used means. Furthermore, the unexploited space in trucks results in huge losses for these companies in addition to the environmental impact, with statistics indicating that 15% to 30% of accidents, traffic congestion, and pollution are caused by empty trucks [Gansterer, Hartl and Vetschera (2019)].

There are many problems that mimic the means of land transport which are presented by researchers to optimise services and minimise their harmful impacts, such as vehicle routing problems, bus scheduling problems [Saha (1970)], cash transportation problems [Yan, Wang and Wu (2012)], railroad blocking problems [Barnhart, Jin and Vance (2000)], and others [Díaz-Parra, Ruiz-Vanoye, Bernábe Loranca et al. (2014); Cornillier, Boctor and Laporte (2008); Zhu, Hu and Wang (2012)].

One of the best-known combinatorial optimisation problems designed to optimise transportation network management and organise the distribution of goods and vehicles is the vehicle routing problem (VRP). The goal of the problem is to distribute goods for a set of customers by finding the best route(s). One or more vehicles with limited capacity from a homogeneous fleet of vehicles are used to transfer goods to customers. Each vehicle must start its trip from a depot and return to it again after serving customers. In the literature, there are different types of VRPs that have been presented over the last 50 years. Although these types vary in constraints and complexity, they share a common goal of minimising the harmful impacts of transport besides reducing cost [Lin, Choy, Ho et al. (2014)].

One important variant of the VRP is the pickup and delivery problem (PDP), which aims to reduce the total transportation cost when people or goods are moved. It differs from the VRP in the type of service provided, with each good or customer transferred between two predefined locations: a pickup node and a delivery node. Several real-world applications of the PDP include the distribution of beverages and the collection of empty bottles and cans, the shipping of cargos and the transportation of raw materials from suppliers to factories. Also, many extensions of the PDP have been presented in the literature that include slight changes to the workings of the original PDP. The selective pickup and delivery problem (SPDP) is one relatively new variant of the PDP that differs from the standard PDPs by serving only some customers rather than serving all. This means the SPDP is valuable to those companies that have limited resources and can realise profits by finding the best routes to serve only the profitable customers. There are two types of SPDPs: (1) SPDPs subject to minimising the travelling cost only (e.g., [Ting and Liao (2013); Liao and Ting (2010); Ho, Sin and Szeto (2016)]) and (2) SPDPs subject to minimising the travelling cost and maximising the profit collection (e.g., [Coelho, Munhoz, Haddad et al. (2012); Bruck, dos Santos and Arroyo (2012)]).

Gansterer et al. [Gansterer, Küçüktepe and Hartl (2017)] presented the multi-vehicle profitable pickup and delivery problem (MVPPDP), which belongs to the second type of SPDP. It aims to find the best routes by minimising the travelling cost and maximising the profit collection in a one-day planning horizon. The MVPPDP is a static problem with a central depot, a set of customers, a predefined number of requests, products, and a homogeneous fleet of vehicles. Each request is defined by a customer pair: pickup customer and delivery customer. Thus, the products are transported from a pickup customer and delivered to a delivery customer to earn a profit from the service. Some real-life applications of the MVPPDP include food delivery mobile apps and delivery companies apps.

Since the MVPPDP is an NP-hard problem (because it is one types of the SPDP that is a special case of a combination of the PDP) [Gansterer, Küçüktepe and Hartl (2017)], metaheuristic algorithms can be utilised to find good solutions within a reasonable processing time, especially for medium and large size problems. In this paper, we proposed an approach that is based on clustering algorithms with proven efficiency in speeding the search process and decreasing the processing time. We used K-means, adaptive K-means and ant colony optimisation (ACO) algorithms to cluster the search space of the MVPPDP. In addition, we modified the greedy randomized adaptive search procedure (GRASP) that was used in Alhujaylan et al. [Alhujaylan and Hosny (2019)] to construct an initial solution for the MVPPDP, after the clustering phase.

The main contributions of this paper are as follows: 1) helping delivery companies and food delivery mobile apps to efficiently plan their services by speeding the selection of customers that can be served and finding the shortest routes to get the profits and achieve customer classification, 2) efficient transport planning can help in decreasing environmentally harmful effects of emissions of CO_2 and other gases besides saving energy resources, and 3) novel solution approaches are proposed to construct an initial solution for the problem which outperform state-of-the-art methods in the literature.

The rest of this paper is organised as follows. A review of some related work is presented in Section 2. Section 3 introduces the formal problem definition. The proposed method is described in detail in Section 4. Section 5 illustrates the experimental results. Finally, conclusions and future work are presented in Section 6.

2 Related work

The MVPPDP belongs to the second type of SPDP that aims to minimise the travelling costs and maximise the profits collected by visiting only the profitable customer pairs. Both the MVPPDP and the profitable tour problem (PTP) share the same goal of finding a good route that maximises the difference between the total collected profit and the total travelling cost. The main difference between them is that in the PTP there are no constrains that must be considered on the vehicle route, such as maximum trip time, vehicle capacity limits, or precedence constraints [Archetti, Speranza and Vigo (2014)]. Studies that address the PTP are rare in the literature [Toth, Paolo and Vigo (2014)]. Therefore, we first present work related to the MVPPDP. Then, some of the studies that have been done on the SPDP with minimising travelling cost and maximising profit collection are briefly presented.

There are five algorithms that have been presented to solve the MVPPDP. The first four algorithms, presented by Küçüktepe et al. [Küçüktepe (2014); Gansterer, Küçüktepe and Hartl (2017)], used the variable neighbourhood search (VNS) metaheuristic. The construction phase of VNS was done using two heuristics: the greedy construction heuristic (C1) and the two-stage cheapest insertion heuristic (C2), while the improvement phase was based on applying the following neighbourhood operators: the pairwise forward exchange, the pairwise backward exchange, relocate pairs, forward insertion, backward insertion, inter tour exchange, inter dummy exchange, inter tour insert, inter dummy insert, and the gravity centre exchange. Authors adapted two strategies for applying the neighbourhood operators: a sequential (Seq) approach that uses a predefined sequence of neighbourhoods and a self-adaptive (Sea) approach that determines the sequence of neighbourhoods by adapting itself according to the search status. Finally, an alternative algorithm that was based on a guided local search (GLS) metaheuristic was implemented to compare with the proposed approach. The proposed algorithm was tested on 36 new randomly generated data instances of different sizes. Experimental results showed that both variants of general VNS outperformed GLS in terms of solution quality.

The fifth algorithm was proposed by Haddad [Haddad (2017)]. Their combination of an iterated local search and random variable neighbourhood descent was proposed to solve the profitable pickup and delivery problem; therefore, they named their algorithm IPPD. The IPPD algorithm was not limited to accepting only feasible solutions during the search. The C1 heuristic that was used in Gansterer et al. [Gansterer, Küçüktepe and Hartl (2017)] was adopted to construct the initial solution. To improve the solution, several of the following neighbourhood moves were then applied: pair swap or pair shift, block swap or block shift, pickup or delivery shift, inter pair swap or inter pair shift, inter block swap or inter block shift, gravity centre exchange, insert operator and remove operator. The proposed algorithm was tested on the benchmark instances proposed by Gansterer et al. [Gansterer, Küçüktepe and Hartl (2017)]. It proved its efficiency in addressing small- and medium-sized instances, for which it was able to find new best solutions for six instances. However, the performance of the proposed algorithm was not adequate for large-sized instances.

Recently Alhujaylan et al. [Alhujaylan and Hosny (2019)] used the construction phase of the well-known GRASP to build initial solutions for the MVPPDP. They compared the methods performance with two construction heuristics that were previously used in the literature to build initial solutions of the MVPPDP. The experimental results proved the proposed method outperformed the other construction heuristics, especially in small-sized instances, where they got eight new best initial solutions for the problem.

One of the practical applications of the SPDP was the distribution of soft drinks and collection of recyclable containers by a Quebec-based company by applying three heuristics as follows: the nearest neighbour heuristic, first petal heuristic, and second petal heuristic. The empirical results indicated that these heuristics contributed to decrease the distance by 23% [Privé, Renaud, Boctor et al. (2006)]. Another application of SPDP is called the single vehicle routing problem with deliveries and selective pickups (SVRPDSP), which has been solved using mixed integer linear programming (MILP) and a Tabu search (TS) algorithm. The proposed algorithm was applied on instances that were derived from the VRP library, and it was able to produce near-optimal solutions for 68 instances [Gribkovskaia, Laporte and Shyshou (2008)]. Additionally, the selective multi-depot vehicle routing problem with pricing required an organised solution to collect cores of durable goods from customers to encourage them to buy new products. A rich neighbourhood TS (TS-RN) coupled with two MILPs (mixed integer linear programming) models was proposed to solve this problem. According to the results, the proposed approach succeeded in terms of both efficiency and accuracy when it was tested on 40 randomly generated instances [Aras, Aksen and Tekin (2011)]. Again, the same instances used in Gribkovskaia et al. [Gribkovskaia, Laporte and Shyshou (2008)] were used again to test two hybrid general variable neighbourhood searches (HGVNSs) and a hybrid metaheuristic based on an evolutionary algorithm (EA) that were proposed by Coelho et al. [Coelho, Munhoz, Haddad et al. (2012); Bruck, dos Santos and Arroyo (2012)] to solve the SVRPDSP. Both metaheuristics were proved to be robust and effective. For the interested reader, other applications of the SPDP can be found in Coelho et al. [Coelho, Munhoz, Ochi et al. (2016); Gutiérrez-Jarpa, Desaulniers, Laporte et al. (2010); Qiu, Feuerriegel and Neumann et al. (2017); Baniamerian, Bashiri and Tavakkoli-Moghaddam (2019); Wen, Larsen, Clausen et al. (2009)].

3 Problem definition

The MVPPDP is a static problem with predefined constituents. We assume that there is a central depot that receives customer requests. These requests consist of pairs of pickup and delivery customers. To serve these requests, we have a set of homogeneous vehicles that transport a number of homogeneous products from a selected (partial) set of pickup customers to their corresponding delivery customers, such that a certain profit is gained. The constraints of the MVPPDP that should be considered are the following:

- *The pairing constraint*: For each request there is a predefined customer pair (pickup and delivery).
- *The precedence constraint*: The pickup customer must be visited before the delivery customer.
- *The trip time constraint*: Each vehicle has a certain daily travel time limit that cannot be exceeded while customers are being served.
- *The capacity constraint*: Each vehicle has a limited capacity that cannot be exceeded, when products are being collected from pickup customers.
- Each vehicle should start/end its journey from/at the depot with an empty load.
- Each customer must be visited only once.

The formal definition of MVPPDP is given by Küçüktepe [Küçüktepe (2014)] as follows:

Let $G = (V, E)$ be a graph, where $V = \{0, \ldots, 2n + 1\}$ is the vertex set. The vertex $(0, 2n + 1)$ represents the central depot. $P = \{1, \ldots, n\}$ is the set of pickup customers, while $D = \{n + 1, \ldots, 2n\}$ is the set of delivery customers. The arc set is $A = \{(i, j): i, j \in V, i \neq j\}$, such that a non-negative routing cost c_{ij} is associated with each arc. It is assumed that a revenue r_i is collected when visiting each delivery vertex i. Also, for a pickup vertex i there is a supply $q_i > 0$, while for a delivery vertex there is a demand $q_{n+i} = -q_i$. It is also assumed that there is no supply or demand for the depot (i.e., $q_0 = 0$). Finally, there is a set of vehicles $k = \{1, \ldots, m\}$, such that each vehicle has a maximum load capacity C and a maximum tour time limit T.
The objective function of the MVPPDP is described as follows:

$$Maximize \sum_{i\in V} \sum_{j\in V} \sum_{k\in K} (r_i - c_{ij}) x_{ijk} \tag{1}$$

where x_{ijk} is a binary decision variable that is equal to one if arc ij is used by vehicle k, and zero otherwise.

For the details of the mathematical model of the MVPPDP, the reader is referred to Gansterer et al. [Gansterer, Küçüktepe and Hartl (2017); Alhujaylan and Hosny (2019)].

4 Proposed method

We divided our proposed method for solving the MVPPDP into two phases: a *clustering phase* and a *routing phase*. The details of each phase are presented below.

4.1 Clustering phase

The purpose of the clustering phase is to try to reduce the search space by dividing customers into clusters based on some relatedness measure. After this, selected customers from each cluster will be visited by one vehicle whose route will be planned in the routing phase. We proposed three clustering algorithms to cluster the MVPPDP search space: 1) a K-means clustering algorithm, 2) an adaptive K-means clustering algorithm, and 3) an ACO-based clustering algorithm. In the MVPPDP, since each request has a pair of customers (pickup and delivery customers), the coordinates of a midpoint between the pickup customer and the delivery customer are computed to represent the request pair as follows:

$$Midpoint\ coordinate\ of\ Request_i = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right) \tag{2}$$

where i represents the index number of a given customer, $i = \{1, \ldots, N\}$, (x_1, y_1) represents the coordinate of the pickup customer, and (x_2, y_2) represents the coordinate of the delivery customer. Thus, the midpoints of requests were considered in all clustering algorithms that have been used. The details of the algorithms are presented in the following sub-sections.

4.1.1 K-means clustering algorithm

The K-means algorithm, which was developed by MacQueen [MacQueen (1967)], is one of the most well-known and simplest unsupervised learning algorithms that have been used to solve the familiar clustering problem. The K-means algorithm has numerous advantages that help make it useable for many clustering problems. For example, it is easy to understand and implement, it has high performance speed, and it is relatively efficient where its time complexity is $O(tKN)$, where t is the number of iterations required for cluster convergence, and N is the number of objects in the dataset. The general idea of the K-means algorithm is to classify a given data set into an a priori number of clusters with each cluster treated as a separate group. Dividing the problem into subproblems in this fashion eases and accelerates the solving of the problem.

A flowchart of the K-means clustering algorithm is illustrated in Fig. 2. In the following steps, we explain how the algorithm works.

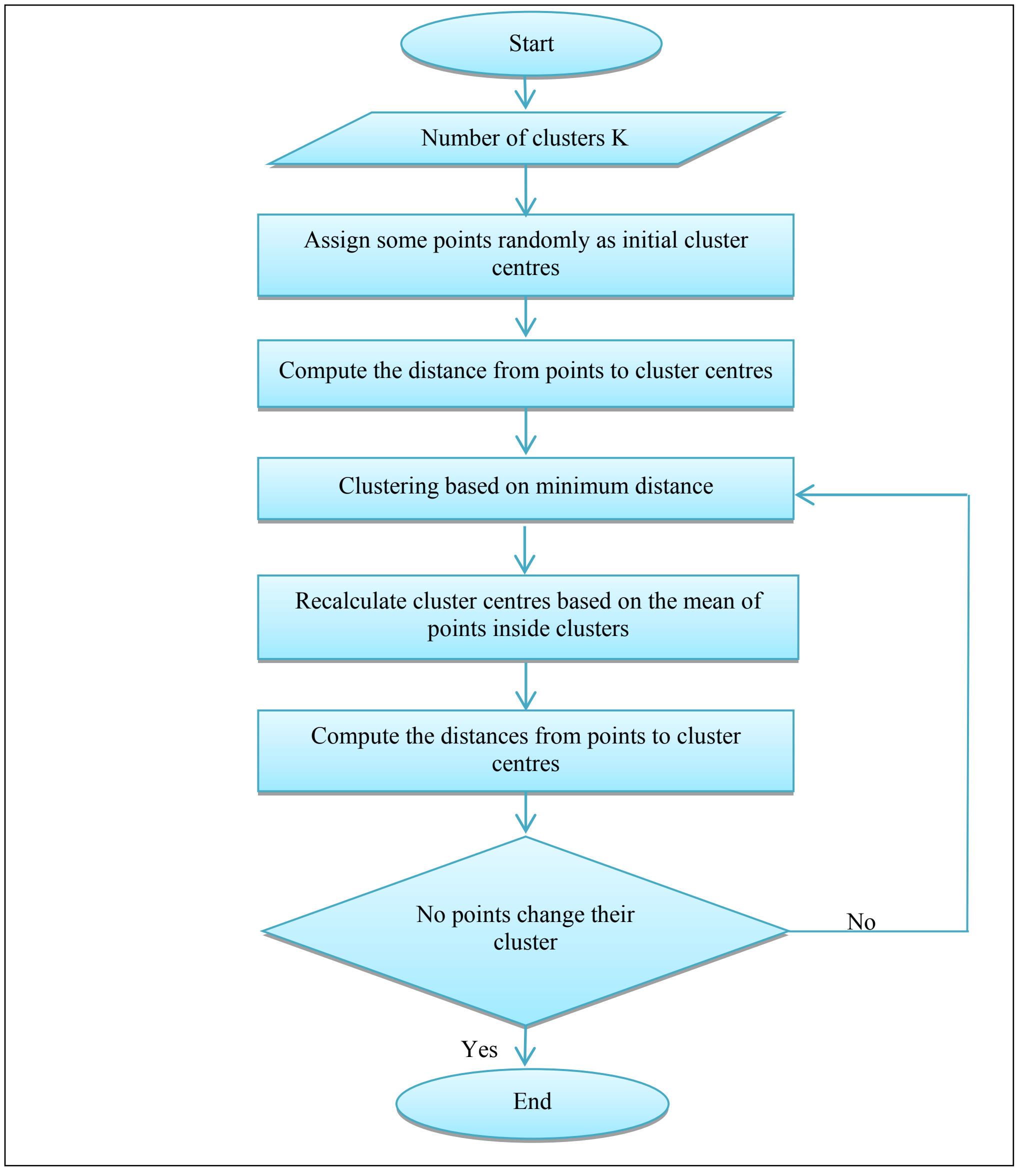
Start
Number of clusters K
Assign some points randomly as initial cluster centres
Compute the distance from points to cluster centres
Clustering based on minimum distance
Recalculate cluster centres based on the mean of points inside clusters
Compute the distances from points to cluster centres
No points change their cluster
No
Yes
End

Figure 2: Flowchart of k-means clustering algorithm

Step 1: Initialisation of parameters
In our K-means algorithm, two initial parameters need to be entered: the number of clusters K and the coordinates of customers.

Step 2: Initial clustering solution
From each cluster $k, k = \{1,2, \dots, K\}$ a random request is selected to be an initial centroid c_k of the cluster. The Euclidean distances between the rest of requests (i.e., those that are not selected as cluster centroid) and the cluster centres $\{c_1, c_2, \dots, c_k\}$ are then computed. Each request is assigned to the nearest cluster centre.

Step 3: Generation of a new clustering solution
For each cluster k, a new cluster centre is calculated by computing the means of the coordinates of all requests that belong to that cluster. We then repeat Step 2, with each request reassigned to the appropriate cluster based on the distance between it and the new cluster centres.

Step 4: Final clustering solution
To generate the final clustering solution, Step 3 is repeated several times until convergence (i.e., until no change is observed between the clustering solution's current iteration and the previous one) [Du (2010)].

4.1.2 Adaptive K-means clustering algorithm

One drawback of the classical K-means algorithm is that it does not take into consideration the particularities of the MVPPDP. Therefore, the main goal of proposing the adaptive K-means clustering algorithm is clustering the search space with consideration of the MVPPDP's objective: maximising profit and minimising travelling cost. In other words, this differs from that of the K-means clustering algorithm by considering profits in addition to the distance between customers. Thus, in addition to the distance between the request and the cluster centre, our adaptive K-means algorithm also considers the "appropriateness" of the request within the cluster. The concept of appropriateness, which was used in Haddad [Haddad (2017)] for inserting new requests into a route, has been adapted to our method as follows: after the centre is computed for each cluster, a request i is assigned to an appropriate cluster based on the following equation:

$$A_{i,c_k} = \frac{r_i}{d_{i,c_k}} \tag{3}$$

where r_i is the revenue of the request i, and d_{i,c_k} is the distance between the request i and the cluster c_k. Thus, each request is assigned to the cluster that has the maximum appropriateness value. The pseudocode for our adaptive K-means clustering algorithm is illustrated in Algorithm 1.

Algorithm 1: Pseudocode for adaptive K-means clustering

```
Input: $K$ (the number of clusters), midpoint coordinates of requests={$1, \dots, N$}
/* Initial clustering solution */
For $k$=1 to $K$
      Choose randomly a request $i$ to be an initial cluster centroid.
End For
•   Compute the distance between the rest of requests and cluster centres.
•   For each request, compute the appropriateness of the request withi n each cluster.
•   Assign each request to the cluster with the maximum appropriateness value.
/* Generating a new clustering solution */
Repeat
    For $k$=1 to $K$
          Reassign a new cluster centre by computing the mean of requests that
          belong to cluster $k$.
    End For
    •   Compute the distance between requests and the new cluster centres.
    •   For each request, compute the appropriateness of the request within each cluster.
    •   Assign each request to the cluster with the maximum appropriateness value.
Until no change between the current clustering solution and the previous one
/* Final clustering solution */
Output: a solution with set of $K$ clusters
```

4.1.3 ACO-based clustering algorithm

Ant colony optimisation (ACO) is a metaheuristic algorithm that mimics the cooperative behaviour of ants foraging in search of food. It was first introduced by Dorigo et al. [Colorni, Dorigo and Maniezzo (1991)].

The ACO algorithm is one of the most well-known of the swarm intelligence algorithms used to solve numerical and combinatorial optimisation problems, and it is particularly useful for problems which require finding the shortest path as a goal (see for example [Dahan, El Hindi, Mathkour et al. (2019); Bell and McMullen (2004)]). Additionally, clustering with ACO or with other models inspired by the behaviour of ants has been used as an alternative to traditional clustering algorithms. See Jafar et al. [Jafar and Sivakumar (2010)] for a survey of interesting clustering approaches based on ant behaviour.

The pseudocode for our MVPPDP clustering-based ACO algorithm is presented in Algorithm 2, where the meaning of each notation is as follows: *Max_Iter*: maximum number of iterations; *Num_Cust:* number of customers; *Pop_Size:* size of population; *K*: number of clusters; and $\tau_{i,k}$: the pheromone trail matrix of size $N * K$, where N refers to number of requests, and K refers to number of clusters.

The details of each step of the clustering using ACO are presented below.

Step 1: Initialisation of parameters

Several parameters need to be set before starting the algorithm, such as, population size (*Pop_Size*), number of clusters (*K)*, number of customers (*Num_Cust)*, maximum number of iterations (*Max_Iter)*, a pheromone trail matrix that is initialised to a low value, and the evaporation rate (ρ).

Step 2: Initial population construction

Initially, each artificial ant starts building its solution by randomly assigning requests into groups (clusters) such that each pair of pickup and delivery customers must belong to the same cluster.

Step 3: Solution evaluation

The quality of each solution is evaluated in terms of the value of the objective function (OF), which aims to achieve high profit at the lowest possible cost. The calculation of the objective function depends on computing the centre of gravity (COG) for each cluster (k), which is used later to indicate the degree of appropriateness of customers within clusters. The concepts of centre of gravity and appropriateness, which were used in [Haddad (2017)] for inserting new requests into a solution, have been adapted for our method as follows: First, the centre of gravity (x_{COG}, y_{COG}) of each cluster k is computed, with the Cartesian coordinates (x_i, y_i) for each request i in the cluster k weighted by the customers revenue r_i:

$$x_{COG(k)} = \frac{\sum_{i \in k} x_i r_i}{\sum_{i \in k} r_i} \tag{4}$$

$$y_{COG(k)} = \frac{\sum_{i \in k} y_i r_i}{\sum_{i \in k} r_i} \tag{5}$$

Then, to determine whether each request i is in the appropriate cluster, as done in the adaptive K-means algorithm, the appropriateness A_i is computed by considering the distance between the customer and the cluster centre with respect to the request's revenue using this equation:

$$A_{i,k} = \frac{r_i}{d_{i,COG(k)}} \tag{6}$$

where $d_{i,COG(k)}$ refers to the distance between request i and the COG of the cluster k.

After this, the OF for each solution is computed as the sum of all appropriateness values for all clusters. Then, the best solution that has the maximum OF value is selected and memorised in the Best_Solutions matrix. The pheromone trail matrix is then updated, as shown in the next step.

Step 4: Pheromone update

The pheromone trail matrix has an important role in improving the quality of solutions during the progress of the algorithm. In our approach, we adopt an offline pheromone update [Talbi (2009)]. Thus, updating the pheromone trail matrix includes two phases:

- ***An evaporation phase:***

To avoid premature convergence and increase the diversification and exploration of the search space, all the values of the pheromone trail matrix ($\tau_{i,k}$) are reduced automatically by a fixed proportion, which is called the evaporation rate (ρ):

$$\tau_{i,k} = (1 - \rho)\tau_{i,k} \tag{7}$$

where the trail value $\tau_{i,k}$ represents the pheromone concentration of request i associated to cluster k.

- ***A reinforcement phase:***

To memorise the characteristics of the best solution (S^*) that was obtained in the current iteration, the values of the pheromone trail matrix for the best solution found are increased by a positive value only for those requests that have been included in the best solution:

$$\tau_{i(S^*),k} = \tau_{i(S^*),k} + \frac{1}{OF(S^*)} \tag{8}$$

Step 5: Generating new solutions

For the process of generating new solutions, we were keen to achieve two goals: first, keeping track of the good solutions that had been obtained in previous iterations, in order to continue searching in those areas. Second, increasing the diversification of solutions to prevent getting stuck in a local optimum solution. To achieve the first goal (intensification), the values in the pheromone matrix helped us to focus on the promising areas that contain good solutions. To achieve the second goal (diversification), we used a simple heuristic of adding some randomness. Thus, each artificial ant constructs its solution using the following strategy:

1. The pheromone trail matrix values are normalised using this equation:
$$p_{i,k} = \frac{\tau_{i,k}}{\sum_{k=1}^{K} \tau_{i,k}} \tag{9}$$
where $p_{i,k}$ is the normalised pheromone probability for the request i belonging to cluster k, and K is the number of cluster s.
2. A random number in the range between 0 and 1 is generated.
3. The request is assigned to the appropriate cluster by comparing the cluster's value in the normalised pheromone matrix to the random number. Thus, the request is assigned to the cluster k if the random number is greater than or equal to the normalized probability of assigning the request to k and less than the probability of assigning it to $k + 1$.
4. Steps 2 and 3 are repeated for all requests to generate a new first solution. Additionally, the same steps are repeated to generate the rest of the new solutions in the population.

After this, the new solutions are evaluated again, and the pheromone matrix is updated as shown in Steps 3 and 4. This process is repeated several times until the stopping criterion is reached, which in our method is reaching the maximum number of iterations.

Step 6: Selecting best solution

After several iterations, the final best clustering solution-the one with the maximum OF -is selected from the Best_Solutions matrix. This clustering solution is the best from among all good solutions obtained at each iteration. Thus, all requests have now been assigned to clusters, and this grouping will be used later in the routing phase to assign a vehicle to visit the customers in each cluster.

Algorithm 2: Pseudocode of the proposed clustering-based ACO

```
Step 1: Initialization of parameters: Pop_Size, K, Num_Cust, Max_Iter, a pheromone trail
matrix $\tau_{i,k}$, and the evaporation rate $\rho$.
Step 2: Constructing initial population
For S=1 to Pop_Size
    Assign the requests randomly to different clusters.
Step 3: Evaluation phase
    Compute the centre of gravity for each cluster.
    Compute the appropriateness of requests within clusters.
    Compute the objective function for each solution.
End For
Select the best solution $S^*$ that has the maximum objective function, and memorize it in
$Best\_Solutions$ matrix.
Step 4: Pheromone update
The pheromone trail matrix of the best solution is updated by:
Evaporation phase: $\tau_{i,k} = (1-\rho)\tau_{i,k}$
Reinforcement Phase: $\tau_{i(S^*),k} = \tau_{i(S^*),k} + 1/OF(S^*)$
Step 5: Generating new solutions
For $m = 1$ to $Max\_iter$
    For S = 1 to Pop_Size
        Normalize the pheromone trail matrix
        For $i = 1$ to $Num\_Cust/2$
            $p_{i,k} = \frac{\tau_{i,k}}{\sum_{k=1}^{K} \tau_{i,k}}$
            Generate a random number $R$
            For $k = 1$ to $K$
                If ($R \geq p_{(i,k)}$ && $R < p_{(i,k+1)}$)
                    Assign the request $i$ to cluster $k$.
                End If
            End For
        End For
    End For
Repeat Step 3 (Evaluation) and Step 4 (Pheromone update)
End For
Step 6: Selecting best solution
Select the best solution in $Best\_Solutions$ matrix to be the final clustering solution.
```

4.2 Routing phase

After the MVPPDP search space has been divided into clusters, with each customer pair placed in the appropriate cluster, we must choose which customer pairs will be served and in what order. In other words, the routing phase starts. However, there are many restrictions to consider before starting the routing phase. First, each cluster must be served by only one vehicle, which means that the number of vehicles used is equal to the number of clusters. Second, each vehicle should start and end its journey from the depot with an empty load. Furthermore, in addition to the pairing, profit, and cost constraints that were considered in the clustering phase, precedence, trip time and vehicle capacity are other constraints that should not be violated when constructing routes. The routing phase in the literature is usually divided into two sub-phases: the solution construction phase and the solution improvement phase. In this paper, we considered just the first sub-phase by proposing a new approach that based on the GRASP as explained next.

GRASP is a multi-start metaheuristic that is commonly applied to solve different combinatorial optimisation problems. It was first introduced by Feo et al. [Feo, Thomas and Resende (1995)]. It consists of two phases: a construction phase and an improvement phase. The construction phase is used to build an initial feasible solution, while the second phase is a local search used to improve the initial solution to get a local optimum. The construction phase of GRASP combines the greedy and randomised features, where the greedy feature selects a set of candidate solutions based on a specific goal (i.e., maximum profit or minimum distance) and sorts the set in what is called the restricted candidate list (RCL), while the randomised feature randomly selects one of the best candidate solutions from the RCL. Combining the two features helps GRASP to be fast, competitive, and able to find quality solutions in a reasonable time. GRASP has successfully contributed to solving multiple variants of VRPs [Layeb, Ammi and Chikhi (2013); Duhamel, Lacomme, Prins et al. (2010); Marinakis (2012)].

In this paper, we used the construction phase of the well-known GRASP to construct an initial solution of the MVPPDP. To increase the chance of getting effective solutions, we used the concept of a population metaheuristic, which creates a population that contains a set of solutions. All these solutions are improved through a number of iterations until the stopping criterion is reached, at which time the best solution is selected. We adopted the methods used in Alhujaylan et al. [Alhujaylan and Hosny (2019)], with several modifications. To distinguish our method from the original, we refer to the method used in Alhujaylan et al. [Alhujaylan and Hosny (2019)] as GRASP while we refer to our new approach that uses clustering algorithms as GRASP with clustering, or GRASP(C). The first difference between the two versions is that GRASP in Alhujaylan et al. [Alhujaylan and Hosny (2019)] constructs the initial solution directly without clustering, which means all customer pairs are candidates for selection, while in GRASP(C) the search space is first clustered, then the initial solutions are constructed for each cluster based on the positions of customer pairs. The rest of modifications are presented as follows.

Selecting seed customers: Each solution contains a set of routes. The number of routes is equal to the number of clusters, with each route served by only one vehicle. Each route in a solution is constructed by first selecting a seed customer based on the computed customer benefit (CB). In GRASP, the CB is calculated by dividing the revenue gained from the delivery customer by the distance between the customers in the pair. By contrast, in GRASP(C) the CB is calculated based on the distance between the depot and the pickup customer. Thus, the pickup customer that is geographically closest to the depot is selected to be the seed customer which is inserted in the route first.

Constructing the routes: To fill the route with unvisited customers, the following process is repeated until either the maximum time allowed for a trip is reached or all unvisited customers have been served. In GRASP [Alhujaylan and Hosny (2019)], all unvisited customer pairs are inserted individually in the best position, such that the best position for the pickup customer is selected before that of the delivery customer, in order to meet the precedence constraint. To clarify the meaning of best position, we present the following example [Alhujaylan and Hosny (2019)]: suppose we have a route $\{0, a, b, -b, -a\}$, where $\{0\}$ is the depot, $\{a, b\}$ are the pickup customers, and $\{-a, -b\}$ are the delivery customers. Assume that an unvisited customer pair $(c, -c)$ is selected for insertion into the best position in this route. To satisfy the precedence constraint, the best position of the pickup customer is assigned first by computing its insertion cost using the equation:

$$Insertion\,cost = c(0,c) + c(c,a) - c(0,a) \tag{10}$$

where c represents the distance cost between two vertices. This equation is applied for all slots in the route, and the position that has the lowest insertion cost is selected for the pickup customer. The same equation is then used to insert the delivery customer with consideration of the position of the pickup customer. After determining the best position for the customer pair, the insertion ratio (IR) is computed as follows:

$$IR_{c,-c} = \frac{r_{c,-c}}{(c_{0,c}+c_{c,a}-c_{0,a}) + (c_{a,-c}+c_{-c,-a}-c_{a,-a})} \tag{11}$$

where r is the revenue of customer pair (c,-c). Using this method, the unvisited customer pairs are inserted into the candidate solution set (CSS) in descending order based on IR values, and then the first elements of the CSS are assigned to the RCL. After that, one unvisited customer pair is selected randomly from the RCL and inserted into the route. This process is repeated for the remaining unvisited customer pairs until either the maximum trip time is reached or no more customers need to be served. This approach, however, is very time consuming; therefore, we modified it in GRASP(C) by changing the method for computing the IR as follows:

$$IR2_{c,-c} = \frac{r_{c,-c}}{c_{(depot,c)}} \tag{12}$$

where $c_{(depot,c)}$ is the distance between the depot and the pickup customer. The unvisited customer pairs are then inserted into the CSS in descending order based on IR2 values, after which we assign the first elements of the CSS to the RCL. Next, one unvisited customer pair is selected randomly from the RCL and inserted into the best position in the route with respect to vehicle capacity, time, and precedence constraints.

Evaluate solution: After constructing all routes, the quality of the solution is evaluated using Eq. (1). The value of the objective function will later be compared to the values for other solutions.

Local best solution: After all the solutions in the population are constructed, the values of their objective functions are compared. The solution with the maximum objective function value is selected to be the local best solution in this iteration.

Final best solution: Again, all the local best solutions selected in each iteration are compared, and the one with the maximum objective function value is chosen to be an initial solution for the MVPPDP.

The outline of the construction phase of our GRASP(C) is presented in Algorithm 3, using the same parameters used in Algorithm 2. The meanings of new notations are as follows: *Num-Clusters*: number of clusters, *CSS:* candidate solutions set, *RCL:* restricted candidate list, *US:* un-served pairs of customers, and *SM:* solutions matrix that contains the best solutions in the population for each iteration.

Algorithm 3: Pseudocode of GRASP(C)

```
For $i$=1 to Max_Iter
    For $S$=1 to Pop_Size
        For $k$=1 to $Num\_Clusters$
            Phase 1: Seed Vertex Selection
            Step 1: Compute the distance between the depot and the pickup customers
            Step 2: Select the pair that has the closet pickup customer to be seed customer
            Phase 2: Route Construction
            While Maximum Route Time is not violated
                Step 3: Compute the insertion ratio (IR2) for all unvisited customers
                Step 4: Put all the candidate customer pairs in the $CSS$ in descending order
                of IR2
                Step 5: Assign half of the candidate customer pairs in the $CSS$ to the $RCL$
                Step 6: Pick one customer pair randomly from the $RCL$ and insert it in its
                best position in the route after checking the precedence, time, and
                capacity constraints
            End While
        End For
        Step 7: Compute the objective function for the solution, and assign the solution to the $SM$
    End For
    Step 8: Select the best solution that has the highest objective function in $SM$ and assign it
    to Final-Best-Solutions matrix
End For
Step 9: Select the best solution that has the highest objective function in the Final-Best-
Solutions matrix to be the initial solution for the $MVPPDP$
```

5 Computational experiments

The computational experiments aim to compare the performance of our proposed algorithm with the construction heuristics: the greedy construction heuristic (C1) and the two-stage cheapest insertion heuristic (C2) [Küçüktepe (2014)], and GRASP [Alhujaylan and Hosny (2019)]. All proposed algorithms have been coded in MATLAB (R2017b) and executed using a laptop computer with an Intel Core i7-4510U CPU @ 2.00 GHz (2601 MHz, two cores, four logical processors). Before we present the results of experiment, the used datasets and parameter tuning details are described in the following sub-sections.

5.1 Test instances

The same instances that were used in Gansterer et al. [Gansterer, Küçüktepe and Hartl (2017)] are used here to test our approach. The data instances are 36 instances that are classified into three groups: small size (20 and 50 customers served by two and three vehicles, respectively), medium size (100 and 250 customers served by four and five vehicles, respectively) and large size (500 and 1000 customers served by six and eight vehicles, respectively). Moreover, each group has 12 instances. These instances diverge from each other with respect to time and revenue. The total time limit is set to be either small or large, with the range within 2500 to 15000 to generate short and long routes. Also, the amounts of revenues are set to be either equal for all customers, proportional to the demands, or random. The revenue is gained from the delivery customer after delivery of goods coming from the pickup customer. The quantity of goods is set to be an integer value from 1 to 50.

5.2 Parameter tuning

5.2.1 ACO parameters tuning

There are only three parameters that need to be tuned in the ACO: size of population, number of iterations, and evaporation rate (ρ). Nine datasets were used to test these parameters: 50 customers with fixed revenue to generate a short route (50-F-S), 50 customers with proportional revenue to generate a short route (50-P-S), 50 customers with random revenue to generate a short route (50-R-S), 250 customers with fixed revenue to generate a short route (250-P-S), 250 customers with proportional revenue to generate a short route (250-P-S), 250 customers with random revenue to generate a short route (250-R-S), 1000 customers with fixed revenue to generate a short route (1000-R-S), 1000 customers with proportional revenue to generate a short route (1000-P-S), and 1000 customers with random revenue to generate a short route (1000-R-S). Since the time constraint is not considered in the clustering phase, we took only instances that generated short routes because the result is the same as that for generating long routes. The details of testing each parameter are as follows.

Population size

Each data instance was tested with different population sizes: 10, 20, 30, and 40. Tab. 1 illustrates the results of tuning the population size. These show that increasing the population size beyond 30 did not lead to an improvement in the OF. Thus, the population size was taken to be 30.

Table 1: Results of tuning the population size of ACO

Dataset Instances	**Population Size**			
	10	**20**	**30**	**40**
50-F-S	6.62	7.77	12.24	12.24
50-P-S	26.59	30.13	47.55	47.55
50-R-S	17.94	24.49	42.23	42.23
250-F-S	10.32	10.32	10.32	10.32
250-P-S	1.53	1.54	1.548	1.548
250-R-S	28.04	28. 26	28.26	28.26
1000-F-S	1.64	1.72	1.72	1.721
1000-P-S	4.85	5.03	5	5
1000-R-S	1.50	1.54	1.63	1.63

Number of iterations

Each data instance was also tested with different numbers of iterations: 100 and 200. Tab. 2 presents the values of the OF after setting the population size to 30, as determined in the previous experiment. The results of testing showed that there was no enhancement in the OF values after 100 iterations. Thus, the maximum number of iterations was taken to be 100.

Table 2: Results of tuning the number of iterations of ACO

Dataset Instances	**Number of Iterations**	
	100	**250**
50-F-S	6.621507	6.621507
50-P-S	26.59067	26.59067
50-R-S	17.94286	17.94286
250-F-S	10.32055	10.32055
250-P-S	1.537698	1.537698
250-R-S	28.04222	28.04222
1000-F-S	1.64904	1.64904
1000-P-S	4.858068	4.858068
1000-R-S	1.505093	1.505093

Evaporation rate

Once again the same instances were used to select a suitable value for the evaporation rate, which is used to increase diversification of the search space and to prevent the algorithm is becoming stuck in local optima. After setting the population size and number of iterations to 30 and 100, respectively, several tests were run with values of ρ=0.2, 0.5, and 0.8. Tab. 3 illustrates that there was no change in OF when the value of ρ was changed.

Table 3: Results of tuning evaporation rate of (ACO)

Dataset Instances	**Evaporation rate (ρ)**		
	0.2	**0.5**	**0.8**
50-F-S	12.24808	12.24808	12.24808
50-P-S	47.55501	47.55501	47.55501
50-R-S	42.23965	42.23965	42.23965
250-F-S	10.32093	10.32093	10.32093
250-P-S	1.548393	1.548393	1.548393
250-R-S	28.26517	28.26517	28.26517
1000-F-S	1.721865	1.721865	1.721865
1000-P-S	5.004374	5.004374	5.004374
1000-R-S	1.631445	1.631445	1.631445

5.2.2 Parameter tuning for GRASP(C)

In GRASP(C), only two parameters need to be tuned: the size of the RCL and the number of iterations (Max_Iter). In our method, we added a third parameter which is the population size (Pop_Size). Empirical experiments were performed to select the most suitable value for each parameter. Since the routing process needs more time than the clustering process, we selected only medium-sized instances with different situations (revenue either equal, proportional to demand, or random and time either long or short) to test these parameters. Six medium-sized data instances were used containing 100 customers served by four vehicles, each with a capacity of 80. The descriptions of these instances are as follows: 100 customers with fixed revenue to generate a short route (100-F-S), 100 customers with fixed revenue to generate a long route (100-F-L), 100 customers with proportional revenue to generate a short route (100-P-S), 100 customers with proportional revenue to generate a long route (100-P-L), 100 customers with random revenue to generate a short route (100-R-S), 100 customers with random revenue to generate a long route (100-R-L). The details of testing each parameter are below.

The restricted candidate list (RCL) size

Since we want to create a population that contains sets of solutions, the diversity of solutions is important. Thus, half of those solutions that are found in the CSS (candidate solution set) were assigned to the RCL; that is, if the number of initial solutions in the CSS was *n*, $\boldsymbol{CSS = n}$ then $\lceil \frac{\boldsymbol{n}}{\boldsymbol{2}} \rceil$ solutions were assigned to the RCL. This value was chosen by trial, because selecting a small RCL resulted in a population of similar solutions.

Number of iterations

The number of iterations was set to be 10, 50, 100, or 500. As seen in Tab. 4, there was no enhancement of OF values once the number of iterations was increased to more than 100.

Table 4: Results of tuning the number of iterations of GRASP(C)

Dataset	**Number of iterations**			
	10	**50**	**100**	**500**
100-F-S	35467.92	35467.92	35467.92	35467.92
100-F-L	52217.92	57389	57389	57389
100-P-S	62858.16	62858.16	64460.88	64460.88
100-P-L	107447.2	107447.2	109137.1	109137.1
100-R-S	62357.97	62357.97	62357.97	62357.97
100-R-L	105606.3	108581.8	111380.2	111380.2

Population Size

Several values of population size were tested: 10, 20, 30, and 40. As seen in Tab. 5, there was no enhancement of OF values once the number of iterations was increased to more than 100. Also, after setting the number of iterations to 100, increasing the population size to more than 30 did not lead to an enhancement of OF values.

Table 5: Results of tuning the population size of GRASP(C)

Dataset	**Population size**			
	10	**20**	**30**	**40**
100-F-S	35467.92	35467.92	35467.92	36037.47
100-F-L	52217.92	57385.29	57569.88	60919.45
100-P-S	62858.16	71005.55	71005.55	65797.14
100-P-L	107447.2	117736.8	117736.8	112448.1
100-R-S	62357.97	65295.61	70818.36	66870.58
100-R-L	105606.3	107314.2	107314.2	101695.6

5.3 Experimental results

In this experiment, we show the results for the MVPPDP using our GRASP(C) approach, in which the initial solution was constructed after clustering the search space using the clustering methods K-means, adaptive K-means, or ACO. Both ACO and GRASP(C) were run five times for every dataset, since they are stochastic approaches, while the K-means and the adaptive K-means were run one time each, because they are deterministic approaches. We compare our results with the GRASP proposed in Alhujaylan et al. [Alhujaylan and Hosny (2019)] (as explained in Section 4.2), and the greedy construction heuristic (C1) and the two-stage cheapest insertion heuristic (C2) presented in Küçüktepe [Küçüktepe (2014)].

Tab. 6 presents our results for the MVPPDP using our methods: K-means-GRASP(C), adaptive K-means-GRASP(C), and ACO-GRASP(C). The results are calculated in terms of the OF of the solution (i.e., the gained profit), which is equal to total revenue minus total travelling cost, as previously shown in Eq. (1). Thus, the larger the OF value, the better the solution obtained. The results of our proposed methods are also compared with the results of C1, C2 [Küçüktepe (2014)], and GRASP [Alhujaylan and Hosny (2019)] in Tab. 6. In the table, the first column presents the name of the instance. The following two columns present the results of the C1 and C2 algorithms in terms of the best OF value. The third through sixth columns show the results of GRASP, K-means-GRASP(C), adaptive K-means-GRASP(C), and ACO-GRASP(C), in terms of the average and best objective values of five runs. For each group of instances of a particular size, the average results are shown in the highlighted row.

Table 6: Comparing construction heuristics' performances in terms of profits

Dataset	C1	C2	GRASP		K-means-GRASP(C)		Adaptive K-means-GRASP(C)		ACO-GRASP(C)	
			Average	Best	Average	Best	Average	Best	Average	Best
1-20-F-S	16185.4	21400	18097.3	18097.3	**30427.58**	**30427.58**	**22329.25**	**22329.25**	17557.29	17557.29
2-20-F-L	22006.2	32400.1	**37937.2**	**37937.3**	**36923.73**	**36923.73**	**39240.95**	**39240.95**	**36322.33**	**36322.33**
3-20-P-S	39646.5	43528	39957.7	39957.7	42589.38	42589.38	42589.38	42589.38	34010.12	34018.63
4-20-P-L	54849.9	55775.5	**57763.9**	**57763.9**	**56671.54**	**56671.54**	**56671.54**	**56671.54**	46346.57	46346.57
5-20-R-S	22954.4	26884	**30039.9**	**30039.9**	**30040**	**30040**	**27498.05**	**27498.05**	24420.96	24420.96
6-20-R-L	30486.7	41933	**42845.9**	**42845.8**	40714.37	40714.37	37610.41	37610.41	37501.81	37501.81
Average-20	31021.5167	36986.7667	**37773.65**	**37773.65**	**39561.1**	**39561.1**	**37656.6**	**37656.6**	32693.18	32694.6
7-50-F-S	15000.5	17958.2	**18396.2**	**18443.6**	**21603.83**	**21603.83**	**21603.83**	**21603.83**	**25123.15**	**25123.15**
8-50-F-L	34988.2	43008.3	32686.3	33259.2	**48392.87**	**50320.08**	**48325.12**	**50320.08**	**48279.06**	**50351.5**
9-50-P-S	53694.8	38796.9	53631.8	53631.8	40566.93	40566.93	42727.95	42727.95	**55643.25**	**55643.25**
10-50-P-L	99109	62731.2	91164.1	92502.1	**120623.3**	**126971.1**	**115622.4**	**117098.7**	**105571.7**	**106544.8**
11-50-R-S	19789.7	24619.2	24364.4	24364.5	**25776.83**	**25776.83**	**25557.18**	**25557.18**	**29969.16**	**29969.16**
12-50-R-L	45326.9	51723.3	**52889.2**	**54712.1**	**75878.26**	**76862.04**	**75116.74**	**76620.08**	**58250.91**	**61689.7**
Average-50	44651.5167	39806.1833	**45522**	**46152.22**	**55473.66**	**57016.8**	**54825.54**	**55654.64**	**53806.21**	**54886.93**
13-100-F-S	19033.1	28818.1	13369.3	13749.3	**35194.98**	**35925.36**	**35308.75**	**36058.97**	25243.62	27213.72
14-100-F-L	31825.6	55322.2	27336.9	28692.9	**58588.92**	**61319.54**	**59725.73**	**61113.85**	45640.54	49279.54
15-100-P-S	44329.1	46547.4	**51218.1**	**54475.1**	**58671.04**	**60896.45**	**58207.45**	**59118.71**	42801.93	42921.71
16-100-P-L	69459.3	79541.3	**80299.3**	**86141.8**	**100866.9**	**102952.5**	**108926**	**116683.5**	**93169.67**	**97599.87**
17-100-R-S	49912.7	70214.6	46160.4	46450.7	58391.44	61503.24	56602.73	61120.87	42829.24	46740.8
18-100-R-L	63116.1	91663.1	76019.9	77136.6	**100849**	**103656.9**	**106575.1**	**112673.5**	86177.11	89791.18
Average -100	45552.64	62017.7833	49067.32	51107.73	**68760.38**	**71042.33**	**70890.96**	**74461.56**	55977.02	58924.47
19-250-F-S	27676.1	40906.1	10831.9	11231.9	32300.89	36215.5	38787.25	39663.81	29917.92	32009.7
20-250-F-L	44456	67845.1	49193.5	50980.9	64273.03	65810.42	**82560.46**	**85651.42**	56683.08	58495.79
21-250-P-S	63930	43247.3	33041.9	34606.8	**71908.14**	**73036.63**	**82494.04**	**91554.54**	**70527.81**	**75400.5**
22-250-P-L	112471	94126.8	102591	107171.1	**131491.2**	**135823.6**	**154525.5**	**159844.5**	**127970.2**	**129315.6**
23-250-R-S	79486.1	102873	40446.5	41914	87701.57	96745.15	95609.78	**100940.6**	78162.92	82512.05
24-250-R-L	130371	143886	128304.5	129344.6	**177237.1**	**184722.9**	**180403.2**	**188565.6**	**157432.1**	**162929.8**
Average-250	71898.3667	83647.3833	60734.88	62541.55	**94151.99**	**98725.69**	**105730**	**111036.7**	**86782.33**	**90110.59**
25-500-F-S	49210	78580.2	21758.3	24012.2	71785.54	76799.28	76091.86	78462.57	53441.02	62680.34
26-500-F-L	73299.8	135652	84616.4	85889.1	120276.4	124711.1	133532.4	**136732.3**	90560.08	93441.86
27-500-P-S	124075	84513.6	49864.8	51297.9	**141348.9**	**148501.2**	**147588**	**152594.3**	106055.3	109622.2
28-500-P-L	179001	169098	138057.2	142740.2	**240201.4**	**251008.2**	**241347.9**	**249010.4**	**183839**	**190679.2**
29-500-R-S	108049	116568	54859.3	56901.7	**168412.9**	**181634.1**	**162507.1**	**166508.9**	**118296.2**	**124116.6**
30-500-R-L	170205	218965	163822.4	166232.5	**273419.2**	**280771.9**	**276322.4**	**282335.3**	**213106.7**	**225958.7**
Average-500	117306.633	133,896.133	85496.4	87845.6	**169240.7**	**177237.6**	**172898.3**	**177607.3**	127549.7	**134416.5**
31-1000-F-S	27655.6	66345.3	9132.4	11045.9	48979.91	54365.44	48644.53	50359.66	20397.7	21662.18
32-1000-F-L	32078.4	112840	56276.1	58068.2	93458.25	96828.26	96498.02	106034.7	47436.16	48585.29
33-1000-P-S	264997	152307	134015.9	141167.4	**285728**	**295025.2**	**284393.9**	**294908.3**	201718.7	208211.4
34-1000-P-L	380514	318268	325103.1	340418.4	**517951.6**	**524767.3**	**505742**	**538695.3**	367480.1	375047.2
35-1000-R-S	193390	197083	99173.9	102051	**252784.2**	**261472.5**	**242126**	**248328.1**	175908.8	181163.4
36-1000-R-L	275805	362266	268887.4	271613.1	**461578.9**	**483610.4**	**439890.4**	**447702.6**	320173.8	326404
Average-1000	195740	201518.217	148764.8	154060.7	**276746.8**	**286011.5**	**269549.1**	**281004.8**	188852.6	193512.2

As can be seen from the results in Tab. 6, all the proposed algorithms demonstrated good performance, on average, in solving the small, medium, and large-sized data instances, compared to previous approaches from the literature. We achieved new best solutions (bold values) for 25 sets of data instances with K-means-GRASP(C), for 26 sets with adaptive-K-means-GRASP(C), and for 13 sets with ACO-GRASP(C). We also found solutions (underlined values) that were better than at least one of the algorithms C1, C2, or GRASP solutions as follows: 11 with K-means-GRASP(C), 10 with adaptive-K-means-GRASP(C), and 13 with ACO-GRASP(C).

On the other hand, when we compared our construction heuristics that rely on first clustering the search space and then constructing the initial solution using GRASP(C), we found that adaptive K-means-GRASP(C) outperformed both K-means-GRASP(C) and ACO-GRASP(C) in the number of new best solutions produced (26 compared to 25 and 13, respectively). Looking at the overall average results for each category, though, we realize that the K-means-GRASP(C) method has a better performance than the other two methods with respect to small size instances (20 and 50 customers), while adaptive K-means-GRASP(C) outperformed the other methods in medium and large size instances (with the exception of instances of size 1000 customers). The reason for this last observation could be that the K-means clustering is based on distance only, which means the customer pairs that are geographically close are grouped together. Since this particular category of instances is very challenging, due to the large number of customers, decreasing the distance between customer pairs, by grouping them in the same cluster makes it possible to serve more customer pairs without exceeding the total trip time, thus increasing the value of the OF.

In addition, looking at the average results for each instance set, the adaptive K-means-GRASP(C) outperformed C1 and C2, while ACO-GRASP(C) demonstrated acceptable performance compared to those heuristics, although it was inferior to both K-means-GRASP(C) and adaptive K-means-GRASP(C). The reason for that could be the approach used in our ACO which has a significant effect on the results. In fact, both K-means and adaptive K-means are based on static computations during the clustering process; K-means clusters customer pairs based on distance, and adaptive K-means clusters them based on distance and revenue. In contrast, ACO clustering is based on a criterion other than the distance and revenue, which is a random insertion heuristic that is used to insert the customer pairs into the appropriate clusters. In other words, for each customer pair, a random number is generated and compared to the pheromone matrix values of the clusters, and the cluster with a value greater than the random number is selected. Thus, the randomness feature of ACO might contribute to inaccurate clustering of customer pairs.

Fig. 3 presents the performance of all proposed construction heuristics compared with C1, C2, and GRASP in terms of profits. It can be observed from this figure that both K-means-GRASP(C) and adaptive K-means-GRASP(C) have comparable results which clearly outperform all other construction heuristics.

On the other hand, we also compared between the construction heuristics based on the execution time (in seconds). Since the execution time is not reported in Küçüktepe et al. [Küçüktepe (2014)], we compared our proposed algorithm with the GRASP of Alhujaylan et al. [Alhujaylan and Hosny (2019)], as shown in Tab. 7. Tab. 7 illustrates that all our clustering algorithms have an effective role in speeding the search process. Also, the modifications that were done on GRASP contribute to making GRASP(C) faster than its predecessor (recall that GRASP selects the best position for each unvisited customer pair then computes their IRs to select one pair randomly in descending order, whereas GRASP(C) first computes the IR2, selects one pair randomly from the first elements in descending order, and then selects the best position for one pair only). Moreover, the time performance of our algorithms, K-means-GRASP(C), adaptive K-means-GRASP(C), ACO-GRASP(C), is comparable, although ACO-GRASP(C) shows slightly shorter processing time.

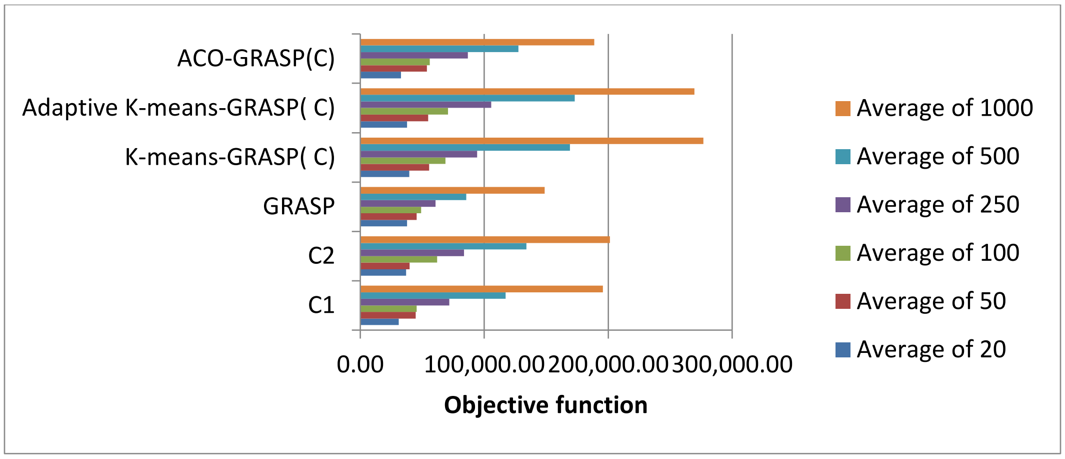

Figure 3: Construction heuristics' performances in terms of average of profits

Therefore, since the three algorithms are comparable with respect to time, we compared between one of them (adaptive K-means-GRASP(C),) and the GRASP of Alhujaylan et al. [Alhujaylan and Hosny (2019)] in terms of time (seconds) as shown in Fig. 4, where the average times for each instance (highlighted rows in Tab. 7) are used to compare their time performances. The huge difference in processing time between our algorithm and the previous GRASP of Alhujaylan et al. [Alhujaylan and Hosny (2019)] is obvious in this figure.

In general, the results in Tabs. 6 and 7 and Figs. 4 and 5 indicate that the proposed algorithms contribute to achieving excellent performance in terms of both quality of solutions and processing time compared with all rival algorithms, GRASP, C1, and C2.

Table 7: Comparison of the results of GRASP, K-means-GRASP(C), adaptive K-means-GRASP(C), and ACO-GRASP(C) in terms of time (seconds)

Dataset	**GRASP**	**ACO-GRASP(C)**	**Adaptive K-means-GRASP(C)**	**K-means-GRASP(C)**
1-20-F-S	15.449	5.90247	6.884987	9.288224
2-20-F-L	27.297	11.1642	10.71063	12.18949

3-20-P-S	12.624	6.20353	6.698972	7.016141
4-20-P-L	31.08	9.74447	11.47247	11.68066
5-20-R-S	18.464	6.79677	6.377124	6.02175
6-20-R-L	27.343	10.8069	9.811818	12.17539
Average-20	22.04283	8.43639	8.66	9.73
7-50-F-S	57.892	11.6429	10.29613	10.19757
8-50-F-L	108.46	21.3455	21.12624	21.36664
9-50-P-S	54.567	10.0189	8.649488	9.228021
10-50-P-L	104.59	17.8235	17.11283	17.4092
11-50-R-S	57.896	10.5924	10.73269	10.43683
12-50-R-L	102.72	17.6683	18.63418	19.97569
Average-50	81.02083	14.8486	14.43	14.77
13-100-F-S	265.74	23.7467	25.8288	25.97908
14-100-F-L	487.07	33.4579	38.44395	39.55519
15-100-P-S	262.4	20.1179	20.79652	23.24088
16-100-P-L	534.28	30.0171	32.60923	35.22743
17-100-R-S	266.37	23.1374	22.63894	22.59382
18-100-R-L	552.56	31.9653	31.4858	35.77533
Average-100	394.7367	27.0737	28.63	30.40
19-250-F-S	2,068	68.1295	66.41826	63.24167
20-250-F-L	9,381.30	98.1781	110.694	94.44695
21-250-P-S	2,033.80	55.4232	60.33002	55.29911
22-250-P-L	8,213.50	84.761	98.99081	85.51775
23-250-R-S	2,331.10	59.5004	63.83406	64.56845
24-250-R-L	8,401.70	90.3252	95.24033	88.4645
Average-250	5,404.90	76.0529	82.58	75.26
25-500-F-S	17,521	145.061	151.3597	155.6808
26-500-F-L	61,818	201.63	211.4198	206.2777
27-500-P-S	16,479	129.102	135.8799	139.7371
28-500-P-L	54,809	177.631	187.6508	191.1451
29-500-R-S	16,931	136.335	151.0996	149.0228
30-500-R-L	56,801	188.597	202.2808	197.4054
Average-500	37,393.17	163.059	173.28	173.21
31-1000-F-S	48,679	425.681	427.7917	426.7781
32-1000-F-L	115,990	582.996	603.3949	584.803
33-1000-P-S	42,604	341.3332	371.9838	368.8414
34-1000-P-L	985,910	475.1481	531.1365	521.5677
35-1000-R-S	52,068	366.4023	404.2715	418.2105
36-1000-R-L	107,973.30	503.1929	558.1441	570.3743
Average-1,000	225,537.40	449.126	482.79	482

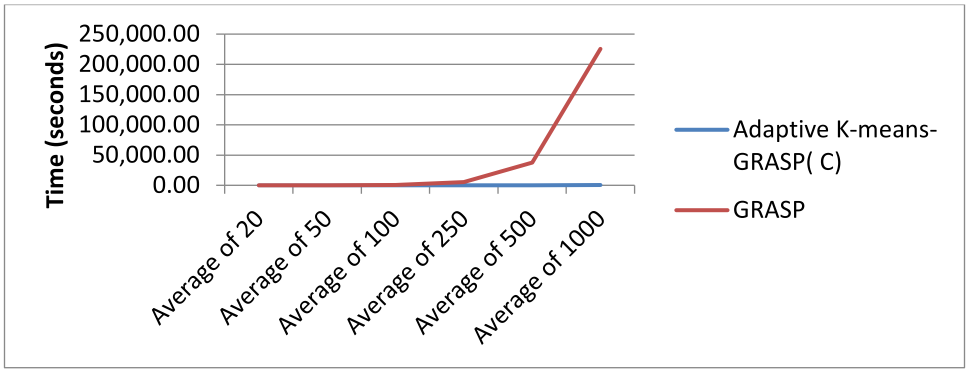

Figure 4: Comparison of the average results of GRASP and adaptive K-means-GRASP(C) in terms of time (seconds)

6 Conclusion

In this paper we presented new heuristics to construct initial solutions for the multi-vehicle profitable pickup and delivery problem (MVPPDP). The heuristics are based on first clustering the search space of the MVPPDP using three clustering methods: K-means, adaptive K-means, and ACO (ant colony optimisation). Then, a modified version of greedy randomised adaptive search procedure (GRASP) has been used to construct the initial MVPPDP solution based on the results of each clustering algorithm. We compared our results with those from the other construction heuristics that have been previously used for the MVPPDP. The experimental results proved the effectiveness of our algorithms in terms of both the solution quality and processing time. The results obtained in our research are beneficial for small-scale pickup and delivery companies with limited resources to improve their planning by reducing their cost and increasing their profit.

In future work, we will improve the initial solutions by using population metaheuristics that combine two features: 1) intensification, to increase the opportunities for getting good solutions, and 2) diversification, to prevent becoming stuck in local optima. Moreover, suitable neighbourhood operators will be carefully selected to be applicable with the constraints of the MVPPDP, thus increasing the chances of producing high-quality solutions.

Acknowledgment: The authors would like to thank Deanship of scientific research for funding and supporting this research through the initiative of DSR Graduate Students Research Support (GSR).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Alhujaylan, A. I.; Hosny, M. I. (2019): A GRASP-based solution construction approach for the multi-vehicle profitable pickup and delivery problem. *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 4, pp. 111-120.

Aras, N.; Aksen, D.; Tuğrul, T. M. (2011): Selective multi-depot vehicle routing problem with pricing. *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 5, pp. 866-884.

Archetti, C.; Speranza, M. G.; Vigo, D. (2014): Chapter 10: vehicle routing problems with profits. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pp. 273-297.

Baniamerian, A.; Bashiri, M.; Tavakkoli-Moghaddam, R. (2019): Modified variable neighborhood search and genetic algorithm for profitable heterogeneous vehicle routing problem with cross-docking, *Applied Soft Computing*, vol. 75, pp. 441-460.

Barnhart, C.; Jin, H.; Vance, P. H. (2000): Railroad blocking: a network design application. *Operations Research*, vol. 48, no. 4, pp. 603-614.

Bell, J. E.; Patrick R. M. (2004): Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, vol. 18, no. 1, pp.41-48.

Bruck, B. P.; dos Santos, A. G.; Arroyo, J. E. C. (2012): Hybrid metaheuristic for the single vehicle routing problem with deliveries and selective pickups. *IEEE Congress on Evolutionary Computation*, pp. 1-8.

Coelho, I. M.; Munhoz, P. L.; Haddad, M. N.; Souza, M. J.; Ochi, L. S. (2012): A hybrid heuristic based on general variable neighborhood search for the single vehicle routing problem with deliveries and selective pickups. *Electronic Notes in Discrete Mathematics*, vol. 39, pp. 99-106.

Coelho, I. M.; Munhoz, P. L.; Ochi, L. S.; Souza, M. J.; Bentes, C. et al. (2016): An integrated CPU-GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups. *International Journal of Production Research*, vol. 54, no. 4, pp. 945-962.

Colorni, A.; Dorigo, M.; Maniezzo, V. (1991): Distributed optimization by ant colonies. *Proceedings of the First European Conference on Artificial Life*, vol. 142, pp. 134-142.

Cornillier, F.; Boctor, F. F.; Laporte, G.; Renaud, J. (2008): An exact algorithm for the petrol station replenishment problem. *Journal of the Operational Research Society*, vol. 59, no. 5, pp. 607-615.

Dahan, F.; El Hindi, K.; Mathkour, H.; AlSalman, H. (2019): Dynamic flying ant colony optimization (DFACO) for solving the traveling salesman problem. *Sensors*, vol. 19, no. 8, pp. 1837.

Díaz-Parra, O.; Ruiz-Vanoye, O.; Bernábe, J. A.; Fuentes-Penna, B.; Fuentes-Penna, A. et al. (2014): A survey of transportation problems. *Journal of Applied Mathematics*, vol. 2014. pp. 1-17.

Du, H. (2010): *Data Mining Techniques and Applications.* An Introduction, Cengage Learning: Andover.

Duhamel, C.; Lacomme, P.; Prins, C.; Prodhon, C.; (2010): A GRASP×ELS approach for the capacitated location-routing problem. *Computers & Operations Research*, vol. 37, no. 11, pp. 1912-1923.

Feo, T. A.; Resende, M. G. (1995): Greedy randomized adaptive search procedures. *Journal of Global Optimization*, vol. 6, no. 2, pp. 109-133.

Gansterer, M.; Hartl, R.; Vetschera, R. (2019): The cost of incentive compatibility in auction-based mechanisms for carrier collaboration. *Networks*, vol. 73, no. 4, pp. 490-514.

Gansterer, M.; Küçüktepe, M.; Hartl, R. F. (2017): The multi-vehicle profitable pickup and delivery problem. *OR Spectrum*, vol. 39, no. 1, pp. 303-319.

Gribkovskaia, I.; Laporte, G.; Shyshou, A. (2008): The single vehicle routing problem with deliveries and selective pickups. *Computers & Operations Research*, vol. 35, no. 9, pp. 2908-2924.

Gutiérrez-Jarpa, G.; Desaulniers, G.; Laporte, G.; Marianov, V. (2010): A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, vol. 206, no. 2, pp. 341-349.

Haddad, M. (2017): *An Efficient Heuristic for One-to-One Pickup and Delivery Problems*. Fluminense Federal Uneversity.

Ho, S. C.; Szeto, W. Y. (2016): GRASP with path relinking for the selective pickup and delivery problem. *Expert Systems with Applications*, vol. 51, pp. 14-25.

Jafar, O. M.; Sivakumar, R. (2010): Ant-based clustering algorithms: a brief survey. *International Journal of Computer Theory and Engineering*, vol. 2, no. 5, pp. 787.

Küçüktepe, M. (2014): *A General Variable Neighbourhood Search Algorithm for the Multi-Vehicle Profitable Pickup and Delivery Problem*. University of Vienna.

Layeb, A.; Ammi, M.; Chikhi, S. (2013): A GRASP algorithm based on new randomized heuristic for vehicle routing problem. *Journal of Computing and Information Technology*, vol. 21, no. 1, pp. 35-46.

Liao, X. L.; Ting, C. K. (2010): An evolutionary approach for the selective pickup and delivery problem. *IEEE Congress on Evolutionary Computation*, pp. 1-8.

Lin, C.; Choy, K. L.; Ho, G. T.; Chung, S. H.; Lam, H. Y. (2014): Survey of green vehicle routing problem: past and future trends. *Expert Systems with Applications*, vol. 41, no. 4, pp. 1118-1138.

MacQueen, J. (1967): Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-297.

Marinakis, Y. (2012): Multiple phase neighborhood search-GRASP for the capacitated vehicle routing problem. *Expert Systems with Applications*, vol. 39, no. 8, pp. 6807-6815.

Privé, J.; Renaud, J.; Boctor, F.; Laporte, G. (2006): Solving a vehicle-routing problem arising in soft-drink distribution. *Journal of the Operational Research Society*, vol. 57, no. 9, pp. 1045-1052.

Qiu, X.; Feuerriegel, S.; Neumann, D. (2017): Making the most of fleets: a profit-maximizing multi-vehicle pickup and delivery selection problem. *European Journal of Operational Research*, vol. 259, no. 1, pp. 155-168.

Saha, J. (1970): An algorithm for bus scheduling problems. *Journal of the Operational Research Society*, vol. 21, no. 4, pp. 463-474.

Talbi, E. G. (2009): *Metaheuristics: From Design to Implementation*. John Wiley & Sons.

Ting, C. K.; Liao, X. L. (2013): The selective pickup and delivery problem: formulation and a memetic algorithm. *International Journal of Production Economics*, vol. 141, no. 1, pp. 199-211.

Toth, P.; Vigo, D. (2014): *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics.

Wen, M.; Larsen, J.; Clausen, J.; Cordeau, J. F.; Laporte, G. (2009): Vehicle routing with cross-docking. *Journal of the Operational Research Society*, vol. 60, no. 12, pp. 1708-1718.

Yan, S.; Wang, S.; Wu, M. W. (2012): A model with a solution algorithm for the cash transportation vehicle routing and scheduling problem. *Computers & Industrial Engineering*, vol. 63, no. 2, pp. 464-473.

Zhu, C.; Hu, J.; Wang, F.; Xu, Y.; Cao, R. (2012): On the tour planning problem. *Annals of Operations Research*, vol. 192, no. 1, pp. 67-86.